LONTAR KOMPUTER VOL. 8, NO. 2, AGUSTUS 2017 **p-ISSN 2088-1541**
DOI : 10.24843/LKJITI.2017.v08.i02.p03 **e-ISSN 2541-5832**

Implementasi *Database Auditing* dengan Memanfaatkan Sinkronisasi DBMS

I Gede Anantaswarya Abhisena [1], I Made Sukarsa [2], Dwi Putra Githa [3]

Program Studi Teknologi Informasi, Fakultas Teknik Universitas Udayana
Bukit Jimbaran, Bali
[1]anantaswaryaas@hotmail.com
[2]e_arsa@ymail.com
[3]dwiputragitha@unud.ac.id

Abstrak

Database auditing dapat menjadi komponen penting dalam keamanan basis data dan kepatuhan terhadap peraturan pemerintah. Database Administrator perlu lebih waspada dalam teknik yang digunakan untuk melindungi data perusahaan, serta memantau dan memastikan bahwa perlindungan yang memadai terhadap data tersedia. Pada tingkat tinggi, database auditing merupakan fasilitas untuk melacak otoritas dan penggunaan sumber daya database. Ketika fungsi auditing diaktifkan, setiap operasi database yang diaudit menghasilkan jejak audit dari perubahan informasi yang dilakukan. Sinkronisasi database adalah bentuk dari replikasi, yang merupakan proses untuk memastikan setiap salinan data pada database berisi objek dan data yang serupa. Sinkronisasi database dapat dimanfaatkan dalam berbagai keperluan, salah satunya membangun auditing untuk mencatat setiap aktivitas yang terjadi pada database. Jejak audit dari operasi database yang dihasilkan, memungkinkan DBA (Database Administrator) memelihara audit trails dari waktu ke waktu, untuk melakukan analisis tentang pola akses dan modifikasi terhadap data pada DBMS (Database Management System).

Kata kunci: *Database Auditing, Jejak Audit, Keamanan Basis Data, Sinkronisasi Data*

Abstract

Database auditing can be an important component in database security and governance regulations. Database Administrator needs to be more alert in the methods used to maintain the corporate data, and standards and requirements available against available data. At high level, databases auditing are facility to search the usage of database authority and resources. When auditing is active, every database operation that audited generates an audit trails of the information changes made. Database synchronization is a form of replication, which is a process to convincing every copy of database comprise the same data. Database synchronization allows to utilized many purposes, one of which builds auditing that records every activities occur at database. The audit trails from the resulting of data operations allows DBAs (Database Administrators) maintained audit trails time to time, to perform an analysis of access patterns and modifications to data in DBMS (Database Management System).

Keywords : *Database Auditing, Audit Trail, Database Security, Data Synchronization*

1. Pendahuluan

Kebutuhan perusahaan dalam melakukan analisis serta pembuatan laporan secara efektif, efisien dan terintegrasi dari sistem informasi menjadi penting untuk dikembangkan [1]. Banyak perusahaan menginginkan proses analisis dilakukan dengan waktu se-*minimum* mungkin [2]. Sistem Informasi telah didefinisikan dan diadopsi ke dalam praktik sejak awal revolusi digital. Sementara informasi telah digunakan untuk proses bisnis, keamanan informasi muncul dalam hal otentikasi dan otorisasi. Konsep tersebut dapat melindungi informasi namun tidak memberikan bantuan dalam penyelidikan. *Log file* diusulkan untuk melacak jejak akses ke *database* dan sistem. Namun, tujuan utama *log file* adalah untuk pemulihan (*recovery*)

transaksi. Untuk dapat menginvestigasi transaksi yang terjadi, maka *database auditing* dapat menjadi pilihan [3]. Melakukan *audit* perubahan data pada *database* sangat penting untuk mengidentifikasi perilaku jahat, menjaga kualitas data, dan meningkatkan kinerja sistem [4].

Database auditing merupakan salah satu masalah utama dalam keamanan informasi. Kurangnya data *auditing* membawa aplikasi bisnis pada hilangnya jejak proses bisnis perusahaan. Untuk membangun *auditing*, data historis atau temporal *database* diperlukan untuk melacak operasi dan tipe operasi dengan waktu [3]. *Database auditing* dapat menjadi komponen penting dalam keamanan basis data dan kepatuhan terhadap peraturan pemerintah. *Database Administrator* perlu lebih waspada dalam teknik yang digunakan untuk melindungi data perusahaan, serta memantau dan memastikan bahwa perlindungan yang memadai terhadap data tersedia [5].

Sinkronisasi *database* adalah bagian dari replikasi, yang merupakan proses untuk memastikan setiap salinan data pada *database* berisi objek dan data yang serupa. Fungsi sinkronisasi yang berjalan pada suatu *database* mengakibatkan data diperbaharui secara *real-time* atau periodik setiap terjadinya perubahan data [6]. Kondisi ini dapat dimanfaatkan untuk membangun *auditing* yang mencatat setiap aktivitas yang terjadi pada *database* bersangkutan.

Kajian terhadap *database auditing* telah dilakukan oleh beberapa penelitian sebelumnya. Beberapa diantaranya memuat teori-teori dalam mendukung proses *auditing*. Penelitian dengan judul *Database Auditing Design on Historical Data* [3] mengusulkan beberapa metode untuk melakukan manajemen historikal data *auditing* pada *database*, seperti *audit* berbasis baris, *audit* berbasis kolom, *audit* dengan tabel log dan *audit* berbasis semi-struktur. Tiga teknik *auditing* pertama dapat diterapkan dengan *database* relasional, sedangkan *audit* berbasis semi-struktur diterapkan dengan menggunakan *extension* dari mesin *database* relasional dan teknologi XML (*extensions markup language*) seperti IBM DB2 9.5, Oracle 10g, dan MS SQL Server. Penelitian dengan judul *Teaching Database Security and Auditing* [7] mengungkapkan banyak jejak *audit* (*audit trails*) yang dihasilkan untuk lingkungan *database*, sehingga terdapat beberapa kategori dalam *auditing.* Kategori *audit* pertama yang dibutuhkan pada kebanyakan lingkungan *auditing* adalah jejak *audit* dari *log on* dan *log off*, serta mencatat semua upaya *log in* yang gagal. Kategori kedua adalah *auditing* terhadap DCL (*Data Control Language*) pada *database*. DCL mencakup perubahan pada hak akses *user*, *user login*, dan atribut keamanan lainnya. Kategori ketiga adalah *auditing* terhadap DDL (*Data Definition Language*) seperti mengubah skema database atau tabel. Beberapa aktivitas pencurian informasi mungkin sering melibatkan perintah DDL. Kategori keempat adalah *auditing* terhadap perubahan data melalui aktivitas DML (*Data Manipulation Language*). Melalui *auditing* pada perintah DML, perubahan yang terjadi, baik nilai lama maupun nilai baru, dapat terekam. Kategori kelima adalah *auditing* perubahan terhadap sumber dari *stored procedure* dan *trigger*, dimana kode program untuk kejahatan dapat dengan mudah disembunyikan. Kategori keenam adalah *auditing* terhadap kesalahan *database* akibat berbagai hal, seperti penyerangan *database* oleh pihak tertentu.

Penelitian ini menerapkan teknik berbasis baris (*row-based auditing*) dalam implementasi *database auditing* untuk melakukan *audit* terhadap aktivitas DML pada transaksi bisnis dengan mempertimbangkan status operasi, waktu yang valid, dan tipe operasi menggunakan model relasional.

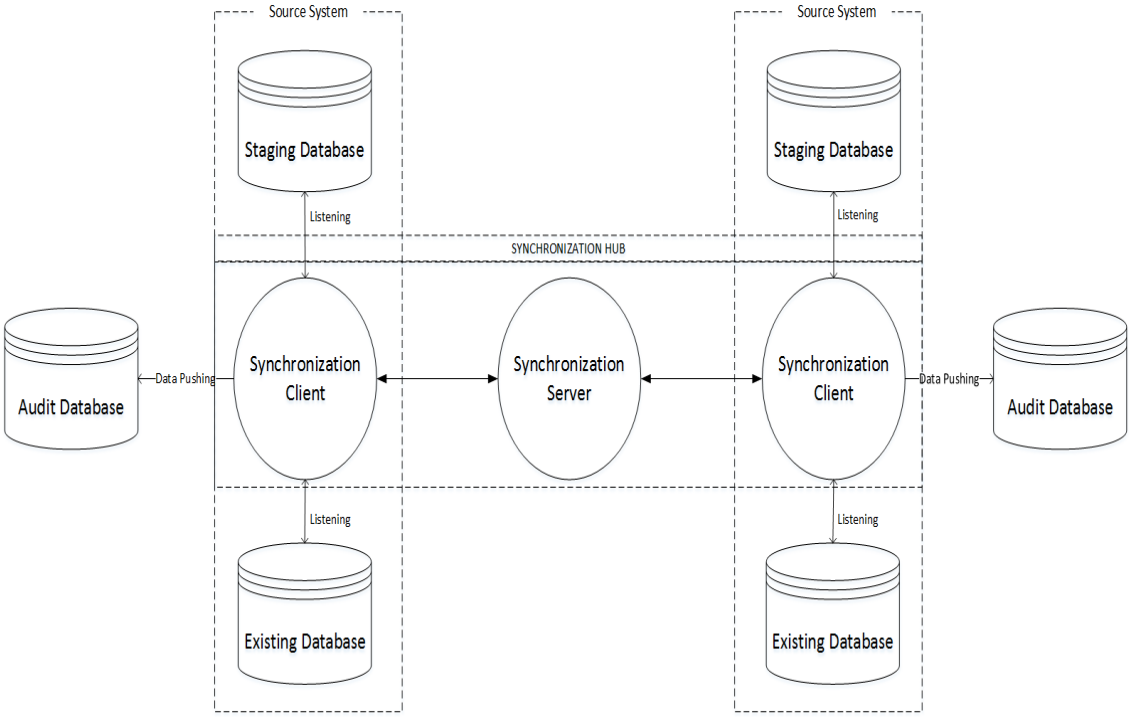
2. Metodologi Penelitian

Penelitian Implementasi *Database Auditing* Dengan Memanfaatkan Sinkronisasi DBMS ini mengunakan metode pengembangan sistem *Waterfall* yang terdapat didalam SDLC (*Software Development Life Cycle*). Pengembangan sistem dengan metode *Waterfall* menjadikan tahap pengembangan menjadi terstruktur. Tahapan analisa penelitian dari Implementasi Database Auditing Dengan Memanfaatkan Sinkronisasi DBMS adalah sebagai berikut :

1. Pendefinisian masalah dari sistem yang dirancang.
2. Pengumpulan data terkait perancangan sistem.
3. Pengumpulan dan penguasaan terkait teori pendukung untuk perancangan sistem.
4. Perancangan *database* menggunakan *platform* MySQL.
5. Perancangan *engine* sinkronisasi menggunakan bahasa pemrograman Python.
6. Pengujian sistem untuk memperoleh hasil yang sesuai.
7. Pengambilan kesimpulan.

2.1. Gambaran Umum Sinkronisasi

Desain sinkronisasi diimplementasikan dalam proses pertukaran data pada DBMS MySQL ditunjukkan pada Gambar 1.



Gambar 1. Skema Arsitektur Sinkronisasi

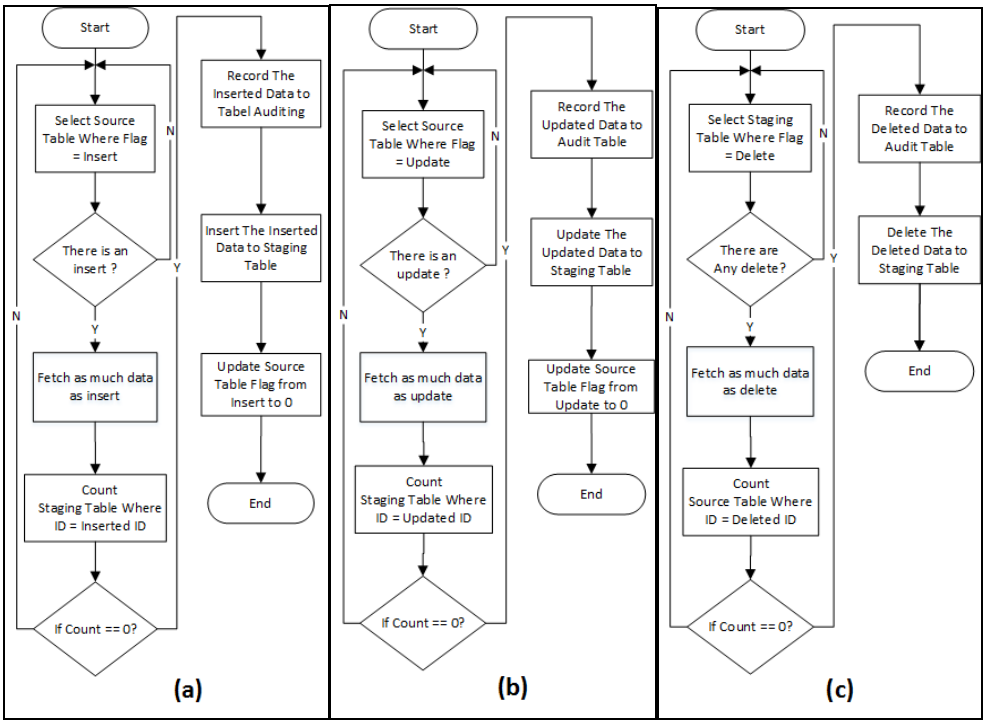
Skema arsitektur dari *engine* sinkronisasi yang dirancang bersifat *real-time*. Setiap terdapat data baru atau terjadinya perubahan data pada *database* eksisting, maka perubahan data tersebut dikirim ke tempat tujuan, tersimpan sebagai *row* baru pada tabel *auditing* yang bersesuaian terhadap transaksi yang telah dilakukan.

Sebelum proses sinkronisasi, *database* yang disinkronkan pada sistem sumber antara *database* eksisting dan *staging* di setiap lokasi harus serupa. Hal ini dilakukan untuk memastikan data yang disinkronkan tetap konsisten. Jika kondisi ini sudah terpenuhi, maka proses sinkronisasi bisa dimulai.

Sinkronisasi yang terjadi pada *database* eksisting dan *database staging* mengakibatkan proses *auditing* menjadi lebih cepat karena *database staging* menempel pada *database* eksisting. Proses *auditing* dilakukan pada sistem sumber, sehingga *auditing database* hanya akan menerima hasil perubahan data yang terjadi.

2.2. Proses *Auditing*

Pada penelitian ini, proses *auditing* dihasilkan dari *engine* sinkronisasi yang bekerja secara terus menerus pada sistem sumber. *Engine* mulai bekerja ketika terjadi *manipulation event* pada *database* eksisting, yaitu ketika *user* memasukkan data baru, mengubah atau menghapus data atau beberapa *field* pada *database* eksisting.



Gambar 2. Alur *Auditing* (a) *Insert*, (b) *Update*, (c) *Delete*

Engine akan bekerja saat terjadi *insert event*, menangkap data yang telah ter-*insert* dan kemudian menyimpannya sebagai *record* baru didalam tabel *auditing* yang sesuai pada *database auditing*. *Update event* untuk satu atau beberapa *field*, membuat *engine* menangkap perubahan yang dibuat dan menyimpannya sebagai *record* baru didalam tabel *auditing* yang sesuai pada *database auditing*. Begitu juga dengan *delete event*, *engine* sinkronisasi akan menangkap data yang terhapus dan menyimpannya sebagai *record* baru pada tabel auditing yang bersesuaian. *Engine* sinkronisasi akan bekerja terhadap tiga kondisi *data manipulation event*, yaitu *insert*, *update*, dan *delete*.

Gambar 2 menjelaskan bagaimana fungsi sinkronisasi pada *engine* bekerja untuk membangun *auditing*. Proses diawali dengan memilih *staging table* yang sesuai dengan table yang mengalami DML *events* pada *database* eksisting. Ketika terjadi perubahan data, maka fungsi akan melakukan *fetching* sebanyak data yang mengalami perubahan, baik *insert*, *update* maupun *delete*. Selanjutnya fungsi akan melakukan *counting* pada tabel sistem sumber berdasarkan ID dari data yang berubah. Jika proses *counting* yang dihasilkan bernilai nol, maka data yang berubah ter-*record* ke table *auditing*.

Membangun *auditing* berbasis sinkronisasi DBMS membutuhkan semua aktivitas DML seperti *insert*, *update*, dan *delete*. *Auditing* berbasis sinkronisasi DBMS memiliki satu kelemahan, dimana tidak semua aktivitas *database* dapat terekam, seperti aktivitas DCL dan DDL. Setiap tabel pada sistem sumber akan dibagi menjadi tiga *events*, masing-masing terdiri dari fungsi *insert*, *update* dan *delete*. Data masukan yang didapat kedalam table *auditing* merupakan data yang dihasilkan dari fungsi sinkronisasi dari setiap event DML yang terjadi pada *database* eksisting.

3. Kajian Pustaka

3.1. *Database Auditing*

Auditing pada dasarnya merupakan kegiatan untuk memonitoring dan merekam kegiatan dari *database* pengguna yang ditentukan. Hasil dari *auditing* yang dihasilkan adalah berupa *audit trail*. Isi dari *audit trail* meliputi catatan yang memberitahu apa saja kejadian yang terjadi pada *database*. Tingkat *record* atau perekaman kejadian yang mampu ditangani setiap DBMS memiliki batasan masing-masing. [8] Meg Coffin Murray menjelaskan bahwa, *database auditing*

dapat digunakan untuk mengidentifikasikan siapa yang mengakses *database*, kegiatan apa yang dilakukan, dan data apa yang diubah.

Meng-*audit* aktivitas dan akses terhadap *database* dapat membantu mengidentifikasi masalah keamanan basis data dan menyelesaikannya dengan cepat. *Auditing* sebagai suatu fungsi, memainkan peran sentral dalam memastikan kepatuhan terhadap aturan karena audit memeriksa dokumentasi tindakan, praktik, dan perilaku bisnis atau individu [7].

Salah satu kunci keberhasilan *auditing* adalah untuk dapat melacak perubahan jejak data, apa operasi modifikasi, dan kapan operasi itu terjadi melalui data historikal. Data historis dapat dimodelkan dalam *database* relasional, dalam beberapa teknik seperti tabel terpisah untuk catatan historis, log transaksi, dan data multi-dimensional. Untuk menjaga historis data dalam *auditing*, beberapa teknik yang disarankan dapat diimplementasikan, seperti *Row-based Auditing* atau *Column-based Auditing* [3].

3.1.1. *Auditing* Dengan *Row-Based*

[3] Teknik ini membuat tabel terpisah dalam setiap tabel relasional untuk menjaga historis data. Tabel operasional tetap sama seperti pada sistem *non-auditing*. Tabel operasional hanya mempertahankan nilai sekarang dari setiap nilai untuk operasi bisnis. Tabel tersebut juga mencakup data statis dan data historis. Data statis tetap tidak berubah atau data yang jarang berubah. Untuk data historis, hanya nilai terakhir yang diperbarui yang akan dipertahankan dalam tabel operasional.

PK	ID	Name	DOB	Address	HiredDate	Salary	StartTime	EndTime	Operation	User
1	101	Tom	1980-01-01	New York	2008-01-01	50000	2008-01-01	2009-01-01	I	Mike
2	101	Tom	1980-01-01	New York	2008-01-01	55000	2009-01-01		U	Mel
3	102	Ann	1985-06-16	London	2009-01-01	60000	2009-01-01		I	Mike
4	103	Jack	1975-01-01	New York	2008-01-01	60000	2008-01-01	2008-06-15	I	Mike
5	103	Jack	1975-01-01	Hong Kong	2008-01-01	60000	2008-06-15	2009-01-01	U	Mike
6	103	Jack	1975-01-01	Hong Kong	2008-01-01	70000	2009-01-01	2009-05-31	U	Mel
7	103	Jack	1975-01-01	Hong Kong	2008-01-01	70000	2009-05-31	2008-05-31	D	Matt

Gambar 3. *Row-based Auditing*

Tabel *auditing* berisi nilai dari setiap kolom tabel operasional seperti yang ditunjukkan pada Gambar 3. Untuk mengurangi operasi *query join*, data statis disertakan dalam tabel *auditing*. Dua *timestamp* waktu dibutuhkan untuk waktu valid, yaitu *start time* dan *end time* untuk mempertahankan umur data. Tipe operasi dicatat untuk mengurangi perbandingan antara historis data yang sama.

3.1.2. *Auditing* Dengan *Column-Based*

[3] *Auditing* berbasis kolom memecahkan redundansi dari *auditing* berbasis baris. Data dalam historis kolom dari tabel *auditing* hanya menyimpan nilai yang berubah kecuali *primary key*, seperti ID, yang digunakan untuk referensi pada tabel operasional. Setiap *record* dalam tabel *auditing* berbasis kolom tidak boleh berisi lebih dari satu nilai data historis karena adanya ketidakpastian terhadap waktu berakhir (*time end*) dari setiap data *auditing.*

PK	ID	Address	Salary	StartTime	EndTime	Opera-tion	User
1	101	New York		2008-01-01		I	Mike
2	101		50000	2008-01-01	2009-01-01	I	Mike
3	101		55000	2009-01-01		U	Mel
4	102	London		2009-01-01		I	Mike
5	102		60000	2009-01-01		I	Mike
6	103	New York		2008-01-01	2008-06-15	I	Mike
7	103		60000	2008-01-01	2009-01-01	I	Mike
8	103	Hong Kong		2008-06-15		U	Mike
9	103		70000	2009-01-01		U	Mel
10	103			2009-05-31		D	Matt

Gambar 4. *Column-based Auditing*

3.2. Sinkronisasi *Database*

Sinkronisasi *database* merupakan proses yang bertujuan menjaga ketetapan atau kosistensi data yang terdapat pada *database server* terhadap data yang berada pada *database server* lainnya. Terdapat fungsi penyalinan data (*copying*) dalam sinkronisasi *database*, yang tersimpan pada suatu tabel *database* lain, baik secara periodik maupun secara *real-time*. Adanya fungsi sinkronisasi *database* memungkinkan perbaharuan data secara *real-time* atau berkala pada *database* yang menjadi objek sinkronisasi. Fungsi sinkronisasi ini merupakan suatu dasar dari adanya replikasi pada DBMS (*Database Management System*) [9].

Sinkronisasi merupakan bagian dari replikasi *database*, merupakan sebuah teknik dalam pendistribusian dan penyalinan data antar *database* sehingga ketetapan atau konsistensi data pada suatu *database* terjaga [10]. Fungsi sinkronisasi memungkinkan pendistribusian data dilakukan secara periodik pada rentang waktu tertentu atau *real-time* ke *host* yang berbeda melalui jaringan komputer. Sinkronisasi *database* dapat mendukung berjalannya fungsi dari aplikasi bisnis, pendistribusian data untuk berbagai keperluan, diantaranya meningkatkan kinerja transaksi bisnis, sistem pengambilan keputusan atau pengolahan sistem terdistribusi pada *server* yang berbeda [9].

3.3. SIDEKA (Sistem Informasi Desa dan Kawasan)

Sistem Informasi Desa dan Kawasan (SIDEKA) merupakan sistem informasi manajemen data desa pada Desa Tangkas, Kabupaten Klungkung, meliputi manajemen data penduduk, data wilayah, dll. Pengaksesan data berupa meng-*entry* data *master* serta melihat berbagai riwayat data.

3.4. *Database Management System* (DBMS)

Database Management System (DBMS) merupakan aplikasi komputer yang memiliki fungsi dalam melakukan manajemen data meliputi proses pemasukkan (*insert*), pengubahan/modifikasi (*update*), penghapusan (*delete*), serta memperoleh data / informasi (*select*) sesuai kebutuhan. Keunggulan dari adanya DBMS sebagai media manajemen data adalah sebagai berikut [9].

1. Praktis: DBMS memberikan fitur media penyimpan data secara permanen dengan ukuran kecil tetapi dapat menyimpan banyak data.
2. Cepat: DBMS sebagai aplikasi komputer dapat mencari dan menampilkan informasi yang dibutuhkan dengan cepat dan akurat.
3. *Up-to-date*: Informasi yang tersedia selalu berubah dan akurat setiap saat.

DBMS dapat di kelompokkan menjadi DBMS homogen dan DMBS heterogen. Sistem DBMS homogen, berisi seluruh *site* menggunakan produk DBMS yang sama. Sedangkan sistem DBMS heterogen, terdiri dari produk DBMS yang beragam, termasuk pada model data yang tidak seragam, sehingga sistem DBMS heterogen terdiri dari beberapa macam model data seperti relasional, jaringan, hirarki dan *object-oriented* DBMS [9].

4. Analisa Hasil dan Pembahasan

Memuat analisa hasil dan pembahasan penelitian yang disajikan dalam bentuk deskripsi penjelasan dari setiap sub bab.

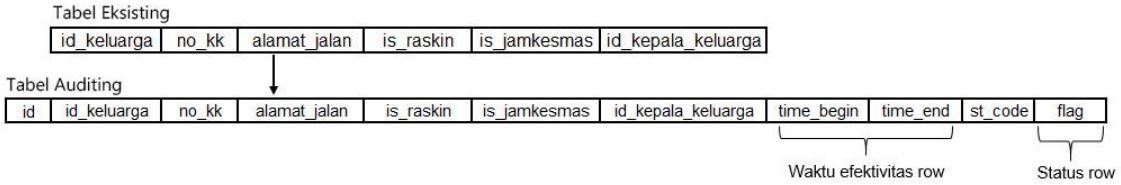
4.1. Analisa Sistem Eksisting

Sistem eksisting yang digunakan sebagai studi kasus *database auditing* adalah SIDEKA (Sistem Informasi Desa dan Kawasan). Sistem ini memiliki dua tabel yang dikelola yaitu, tabel keluarga dan tabel penduduk yang menjadi objek dalam proses *auditing*.

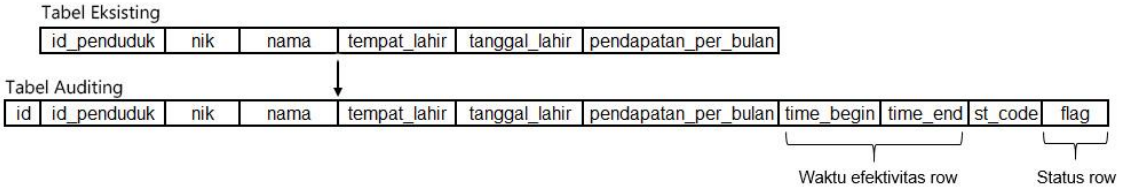
4.2. Desain *Auditing*

Berdasarkan beberapa tipe *database auditing* yang ada, penelitian ini menerapkan model *Row-based Auditing*. Model ini menggunakan tabel *auditing* terpisah dari tabel operasional yang ada (eksisting) untuk melakukan proses *auditing*. Tabel *auditing* berisi nilai dari setiap kolom pada

tabel eksisting yang disajikan secara historis dengan tambahan beberapa atribut diantaranya flag, time begin, time end, dan st code seperti yang ditunjukkan pada Gambar 5 dan Gambar 6.



Gambar 5. Desain *Auditing* Tabel Keluarga



Gambar 6. Desain *Auditing* Tabel Penduduk

Model *row-based auditing* ini menyederhanakan pelaksanaan dari proses *audit*. Ketika pernyataan DML seperti *insert*, *update*, dan *delete* dijalankan pada tabel operasional, engine sinkronisasi hanya dapat menyalin setiap nilai dalam catatan ke dalam tabel *auditing*. Pada saat yang sama, kolom "time_end" dari historis sebelumnya harus diperbarui bersamaan dengan waktu operasi terjadi.

4.3. Pengujian Hasil

Pengujian *auditing* ini dilakukan dengan proses manipulasi data pada *database* eksisting SIDEKA (Sistem Informasi Desa dan Kawasan) yang mempengaruhi *database staging* dalam proses sinkronisasi dan mengirim perubahan data ke *database auditing*. Untuk pengujian proses *insert*, pengujian dilakukan dengan memasukkan data pada tabel keluarga melalui aplikasi DBMS *client* SQLyog seperti yang ditunjukan pada Gambar 7 dan Gambar 8 berikut.

id_keluarga	no_kk	alamat_jalan	is_raskin	is_jamkesmas	id_kepala_keluarga
1	5105031807072456	DUSUN PEKEN DESA TANGKAS	N	N	1
2	5105031807072500	DUSUN TUSAN DESA TANGKAS	N	N	5
3	5105032606120001	DUSUN PEKEN DESA TANGKAS	N	N	11
4	5105030102100004	DUSUN AMBENGAN DESA TANGKAS	N	N	13
5	5105031807072496	DUSUN AMBENGAN DESA TANGKAS	N	N	15
6	5105031807072497	DUSUN TUSAN DESA TANGKAS	N	N	18
7	5105031807072498	DUSUN MERANGGEN DESA TANGKAS	N	N	20
8	5105031807072549	DUSUN MERANGGEN DESA TANGKAS	N	N	24
9	5105031807072484	DUSUN PEKEN DESA TANGKAS	N	N	30
368	5105031807072438	DUSUN TUSAN DESA TANGKAS	N	N	15

Gambar 7. Hasil *Insert* Pada Tabel Keluarga Eksisting

id_penduduk	nik	nama	tempat_lahir	tanggal_lahir	pendapatan_per_bulan
1	5105030710700001	I WAYAN NADRASH	TANGKAS	1970-10-07 00:00:00	3000000
5	5105030107600004	I WAYAN TIKA SH	KLUNGKUNG	1960-12-12 00:00:00	1200000
7	5105032004880001	I GEDE JUSTIKA KORI	KLUNGKUNG	1988-04-20 00:00:00	1500000
8	5105030610910001	I MADE DWIJA PARAMARTHA KORI	KLUNGKUNG	1991-10-06 00:00:00	1500000
9	5105030107380060	I KETUT WANDRIS	TANGKAS	1938-12-31 00:00:00	2000000
10	5105034107380087	NI WAYAN ASIH	TANGKAS	1938-12-31 00:00:00	500000
11	5115133112400089	I NENGAH KAYUN	TANGKAS	1940-12-31 00:00:00	2500000
13	5105033112470074	I NENGAH TISTA	TANGKAS	1947-12-31 00:00:00	1500000
14	5105037112510098	NI WAYAN PURNI	GELGEL	1970-01-01 00:00:00	1300000
15	5105033112720022	I KETUT SUCANA	TANGKAS	1972-12-31 00:00:00	1700000
1340	5105033112728742	I MADE BELAYU	TANGKAS	1988-06-14 00:00:00	2000000

Gambar 8. Hasil *Insert* Pada Tabel Penduduk Eksisting

Hasil *insert* pada tabel keluarga ditunjukkan pada *row* dengan "id_keluarga" bernilai "368", serta hasil *insert* pada tabel penduduk dengan "id_penduduk" bernilai "1340". Saat proses *insert* terjadi pada tabel eksisting, *engine* akan mengidentifikasi proses DML yang terjadi dan memicu engine untuk melakukan sinkronisasi pada tabel yang bersesuaian didalam *database staging* serta menyimpan perubahan data kedalam tabel *auditing.* Gambar 9 menunjukkan *engine* menangkap event DML dari proses *insert* yang dilakukan pada tabel keluarga.

Gambar 9. *Engine* Bekerja Menangkap Perubahan *Insert* Pada Tabel Eksisting

Dalam waktu yang bersamaan, *engine* akan me-*record* aktivitas DML *insert* yang terjadi pada tabel keluarga dan tabel penduduk lalu menyimpannya sebagai *row* baru didalam tabel *auditing* beserta beberapa atribut tambahan diantaranya *field* "time_begin" dan "time_end" untuk mengetahui umur atau efektivitas dari suatu *row*/data, *field* "st_code" untuk mengidentifikasi proses DML yang terjadi dari data/*row* tersebut serta *field* "flag" untuk mengetahui data aktif/non-aktif.

id	id_keluarga	no_kk	alamat_jalan	is_raskin	is_jamkesmas	id_kepala_keluarga	time_begin	time_end	st_code	flag
437	1	5105031807072456	DUSUN PEKEN DESA TANGKAS	N	N	1	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
438	2	5105031807072500	DUSUN TUSAN DESA TANGKAS	N	N	5	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
439	3	5105032606120001	DUSUN PEKEN DESA TANGKAS	N	N	11	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
440	4	5105030102100004	DUSUN AMBENGAN DESA TANGKAS	N	N	13	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
441	5	5105031807072496	DUSUN AMBENGAN DESA TANGKAS	N	N	15	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
442	6	5105031807072497	DUSUN TUSAN DESA TANGKAS	N	N	18	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
443	7	5105031807072498	DUSUN MERANGGEN DESA TANGKAS	N	N	20	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
444	8	5105031807072549	DUSUN MERANGGEN DESA TANGKAS	N	N	24	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
445	9	5105031807072484	DUSUN PEKEN DESA TANGKAS	N	N	30	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
446	368	5105031807072438	DUSUN TUSAN DESA TANGKAS	N	N	15	2017-06-07 15:24:36	0000-00-00 00:00:00	I	1

Gambar 10. Hasil Proses *Insert Auditing* Untuk Tabel Keluarga

id	id_penduduk	nik	nama	tempat_l...	tanggal_lahir	pendapatan_per_bulan	time_begin	time_end	st_code	flag
1372	1	5105030710700001	I WAYAN NADRASH	TANGKAS	1970-10-07 00:00:00	3000000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1373	5	5105030107600004	I WAYAN TIKA SH	KLUNGKUNG	1960-12-12 00:00:00	1200000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1374	7	5105032004880001	I GEDE JUSTIKA KORI	KLUNGKUNG	1988-04-20 00:00:00	1500000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1375	8	5105030610910001	I MADE DWIJA PARAMARTHA KORI	KLUNGKUNG	1991-10-06 00:00:00	1500000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1376	9	5105030107380060	I KETUT WANDRIS	TANGKAS	1938-12-31 00:00:00	2000000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1377	10	5105034107380087	NI WAYAN ASIH	TANGKAS	1938-12-31 00:00:00	500000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1378	11	5115133112400089	I NENGAH KAYUN	TANGKAS	1940-12-31 00:00:00	2500000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1379	13	5105033112470074	I NENGAH TISTA	TANGKAS	1947-12-31 00:00:00	1500000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1380	14	5105037112510098	NI WAYAN PURNI	GELGEL	1970-01-01 00:00:00	1300000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1381	15	5105033112720022	I KETUT SUCANA	TANGKAS	1972-12-31 00:00:00	1700000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1382	1340	5105033112728742	I MADE BELAYU	TANGKAS	1988-06-14 15:26:39	2000000	2017-06-07 15:27:04	0000-00-00 00:00:00	I	1

Gambar 11. Hasil Proses *Insert Auditing* Untuk Tabel Penduduk

Gambar 10. dan Gambar 11. menunjukkan data hasil *insert* pada tabel keluarga eksisting dengan "id_keluarga" = "368" dan pada tabel penduduk dengan "id_penduduk" = "1340", telah tersimpan sebagai historis data didalam tabel *auditing*. Pengujian selanjutnya yaitu *event* DML *update* yang akan di ujicoba pada tabel yang sama yaitu pada tabel keluarga dan tabel penduduk seperti yang ditunjukkan pada Gambar 12 dan Gambar 13.

id_keluarga	no_kk	alamat_jalan	is_raskin	is_jamkesmas	id_kepala_keluarga
1	5105031807072456	DUSUN PEKEN DESA TANGKAS	N	N	1
2	5105031807072500	DUSUN TUSAN DESA TANGKAS	N	N	5
3	5105032606120001	DUSUN PEKEN DESA TANGKAS	N	N	11
4	5105030102100004	DUSUN AMBENGAN DESA TANGKAS	N	N	13
5	5105031807072496	DUSUN AMBENGAN DESA TANGKAS	N	N	15
6	5105031807072497	DUSUN TUSAN DESA TANGKAS	N	N	18
7	5105031807072498	DUSUN MERANGGEN DESA TANGKAS	N	N	20
8	5105031807072549	DUSUN MERANGGEN DESA TANGKAS	N	N	24
9	5105031807072484	DUSUN PEKEN DESA TANGKAS	N	N	30
368	5105031807072438	DUSUN AMBENGAN DESA TANGKAS	Y	Y	15

Gambar 12. Hasil *Update* Pada Tabel Keluarga Eksisting

id_penduduk	nik	nama	tempat_lahir	tanggal_lahir	pendapatan_per_bulan
1	5105030710700001	I WAYAN NADRASH	TANGKAS	1970-10-07 00:00:00	3000000
5	5105030107600004	I WAYAN TIKA SH	KLUNGKUNG	1960-12-12 00:00:00	1200000
7	5105032004880001	I GEDE JUSTIKA KORI	KLUNGKUNG	1988-04-20 00:00:00	1500000
8	5105030610910001	I MADE DWIJA PARAMARTHA KORI	KLUNGKUNG	1991-10-06 00:00:00	1500000
9	5105030107380060	I KETUT WANDRIS	TANGKAS	1938-12-31 00:00:00	2000000
10	5105034107380087	NI WAYAN ASIH	TANGKAS	1938-12-31 00:00:00	500000
11	5115133112400089	I NENGAH KAYUN	TANGKAS	1940-12-31 00:00:00	2500000
13	5105033112470074	I NENGAH TISTA	TANGKAS	1947-12-31 00:00:00	1500000
14	5105037112510098	NI WAYAN PURNI	GELGEL	1970-01-01 00:00:00	1300000
15	5105033112720022	I KETUT SUCANA	TANGKAS	1972-12-31 00:00:00	1700000
1340	5105033112728742	I MADE BELAYU	TANGKAS	1988-06-14 00:00:00	3000000

Gambar 13. Hasil *Update* Pada Tabel Penduduk Eksisting

Hasil *update* pada tabel keluarga ditunjukkan pada *row* dengan “id_keluarga” bernilai “368”, serta hasil *update* pada tabel penduduk dengan “id_penduduk” bernilai “1340”. Proses *update* pada tabel keluarga dilakukan dengan memaipulasi beberapa *field* yaitu “alamat_jalan”, “is_raskin” dan “is_jamkesmas” dengan masing-masing nilai “DUSUN AMBENGAN DESA TANGKAS”, “Y”, dan “Y”. Sementara itu proses *update* pada tabel penduduk dilakukan dengan memaipulasi *field* “pendapatan_per_bulan” dengan nilai “3000000” melalui aplikasi diatas.

Saat proses *update* terjadi pada tabel eksisting, *engine* akan mengidentifikasi proses DML yang terjadi dan memicu engine untuk melakukan sinkronisasi pada tabel yang bersesuaian didalam *database staging* serta menyimpan perubahan data kedalam tabel *auditing.*

Gambar 14. *Engine* Bekerja Menangkap Perubahan *Update* Pada Tabel Eksisting

Update event yang terjadi pada tabel eksisting memicu *engine* untuk merekam perubahan data yang terjadi pada sistem sumber kedalam tabel *auditing* seperti yang ditampilkan pada Gambar 15 dan Gambar 16.

id	id_keluarga	no_kk	alamat_jalan	is_raskin	is_jamkesmas	id_kepala_keluarga	time_begin	time_end	st_code	flag
437	1	5105031807072456	DUSUN PEKEN DESA TANGKAS	N	N	1	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
438	2	5105031807072500	DUSUN TUSAN DESA TANGKAS	N	N	5	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
439	3	5105032606120001	DUSUN PEKEN DESA TANGKAS	N	N	11	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
440	4	5105030102100004	DUSUN AMBENGAN DESA TANGKAS	N	N	13	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
441	5	5105031807072496	DUSUN AMBENGAN DESA TANGKAS	N	N	15	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
442	6	5105031807072497	DUSUN TUSAN DESA TANGKAS	N	N	18	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
443	7	5105031807072498	DUSUN MERANGGEN DESA TANGKAS	N	N	20	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
444	8	5105031807072549	DUSUN MERANGGEN DESA TANGKAS	N	N	24	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
445	9	5105031807072484	DUSUN PEKEN DESA TANGKAS	N	N	30	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
446	368	5105031807072438	DUSUN TUSAN DESA TANGKAS	N	N	15	2017-06-07 15:24:36	2017-06-07 15:54:29	I	0
448	368	5105031807072438	DUSUN AMBENGAN DESA TANGKAS	Y	Y	15	2017-06-07 15:54:29	0000-00-00 00:00:00	U	1

Gambar 15. Hasil Proses *Update Auditing* Untuk Tabel Keluarga

id	id_penduduk	nik	nama	tempat_l...	tanggal_lahir	pendapatan_per_bulan	time_begin	time_end	st_code	flag
1372	1	5105030710700001	I WAYAN NADRASH	TANGKAS	1970-10-07 00:00:00	3000000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1373	5	5105030107600004	I WAYAN TIKA SH	KLUNGKUNG	1960-12-12 00:00:00	1200000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1374	7	5105032004880001	I GEDE JUSTIKA KORI	KLUNGKUNG	1988-04-20 00:00:00	1500000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1375	8	5105030610910001	I MADE DWIJA PARAMARTHA KORI	KLUNGKUNG	1991-10-06 00:00:00	1500000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1376	9	5105030107380060	I KETUT WANDRIS	TANGKAS	1938-12-31 00:00:00	2000000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1377	10	5105034107380087	NI WAYAN ASIH	TANGKAS	1938-12-31 00:00:00	500000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1378	11	5115133112400089	I NENGAH KAYUN	TANGKAS	1940-12-31 00:00:00	2500000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1379	13	5105033112470074	I NENGAH TISTA	TANGKAS	1947-12-31 00:00:00	1500000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1380	14	5105037112510098	NI WAYAN PURNI	GELGEL	1970-01-01 00:00:00	1300000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1381	15	5105033112720022	I KETUT SUCANA	TANGKAS	1972-12-31 00:00:00	1700000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1382	1340	5105033112728742	I MADE BELAYU	TANGKAS	1988-06-14 00:00:00	2000000	2017-06-07 15:27:04	2017-06-07 16:06:33	I	0
1383	1340	5105033112728742	I MADE BELAYU	TANGKAS	1988-06-14 00:00:00	3000000	2017-06-07 16:06:33	0000-00-00 00:00:00	U	1

Gambar 16. Hasil Proses *Update Auditing* Untuk Tabel Penduduk

Gambar 15. dan Gambar 16. menunjukkan data hasil *update* pada tabel keluarga dengan “id_keluarga” = “368” dan pada tabel penduduk dengan “id_penduduk” = “1340”, tersimpan secara historis sebagai *row* baru didalam tabel *auditing*. Data hasil *update* pada tabel eksisting menunjukan bahwa data tersebut bernilai aktif menggantikan rekaman data hasil *insert* pada proses sebelumnya, yang ditunjukkan dengan *field* “flag” bernilai “1”.

Pengujian selanjutnya yaitu event DML *delete* yang akan di ujicoba pada tabel yang sama yaitu pada tabel keluarga dan tabel penduduk seperti yang ditampilkan pada Gambar 17 dan Gambar 18.

id_keluarga	no_kk	alamat_jalan	is_raskin	is_jamkesmas	id_kepala_keluarga
1	5105031807072456	DUSUN PEKEN DESA TANGKAS	N	N	1
2	5105031807072500	DUSUN TUSAN DESA TANGKAS	N	N	5
3	5105032606120001	DUSUN PEKEN DESA TANGKAS	N	N	11
4	5105030102100004	DUSUN AMBENGAN DESA TANGKAS	N	N	13
5	5105031807072496	DUSUN AMBENGAN DESA TANGKAS	N	N	15
6	5105031807072497	DUSUN TUSAN DESA TANGKAS	N	N	18
7	5105031807072498	DUSUN MERANGGEN DESA TANGKAS	N	N	20
8	5105031807072549	DUSUN MERANGGEN DESA TANGKAS	N	N	24
9	5105031807072484	DUSUN PEKEN DESA TANGKAS	N	N	30

Gambar 17. Hasil *Delete* Pada Tabel Keluarga Eksisting

id_penduduk	nik	nama	tempat_lahir	tanggal_lahir	pendapatan_per_bulan
1	5105030710700001	I WAYAN NADRASH	TANGKAS	1970-10-07 00:00:00	3000000
5	5105030107600004	I WAYAN TIKA SH	KLUNGKUNG	1960-12-12 00:00:00	1200000
7	5105032004880001	I GEDE JUSTIKA KORI	KLUNGKUNG	1988-04-20 00:00:00	1500000
8	5105030610910001	I MADE DWIJA PARAMARTHA KORI	KLUNGKUNG	1991-10-06 00:00:00	1500000
9	5105030107380060	I KETUT WANDRIS	TANGKAS	1938-12-31 00:00:00	2000000
10	5105034107380087	NI WAYAN ASIH	TANGKAS	1938-12-31 00:00:00	500000
11	5115133112400089	I NENGAH KAYUN	TANGKAS	1940-12-31 00:00:00	2500000
13	5105033112470074	I NENGAH TISTA	TANGKAS	1947-12-31 00:00:00	1500000
14	5105037112510098	NI WAYAN PURNI	GELGEL	1970-01-01 00:00:00	1300000
15	5105033112720022	I KETUT SUCANA	TANGKAS	1972-12-31 00:00:00	1700000

Gambar 18. Hasil *Delete* Pada Tabel Penduduk Eksisting

Proses *delete* pada tabel keluarga dilakukan pada data dengan “id_keluarga” bernilai “368” seperti yang ditunjukkan pada Gambar 12, serta pada tabel penduduk pada data dengan “id_penduduk” bernilai “1340” seperti yang ditunjukkan pada Gambar 13. Saat proses *delete* terjadi pada tabel eksisting, fungsi pada *engine* akan mengidentifikasi proses DML yang terjadi

dan memicu *engine* untuk melakukan sinkronisasi pada tabel yang bersesuaian didalam *database staging* serta menyimpan perubahan data kedalam tabel *auditing* sebagai *row* baru.

Gambar 19. *Engine* Bekerja Menangkap Perubahan *Delete* Pada Tabel Eksisting

Proses *delete* yang terjadi pada tabel keluarga dan tabel penduduk ter-*record* sebagai *row* baru dalam table *auditing* seperti yang ditampilkan pada Gambar 20 dan Gambar 21.

id	id_keluarga	no_kk	alamat_jalan	is_raskin	is_jamkesmas	id_kepala_keluarga	time_begin	time_end	st_code	flag
437	1	5105031807072456	DUSUN PEKEN DESA TANGKAS	N	N	1	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
438	2	5105031807072500	DUSUN TUSAN DESA TANGKAS	N	N	5	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
439	3	5105032606120001	DUSUN PEKEN DESA TANGKAS	N	N	11	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
440	4	5105030102100004	DUSUN AMBENGAN DESA TANGKAS	N	N	13	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
441	5	5105031807072496	DUSUN AMBENGAN DESA TANGKAS	N	N	15	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
442	6	5105031807072497	DUSUN TUSAN DESA TANGKAS	N	N	18	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
443	7	5105031807072498	DUSUN MERANGGEN DESA TANGKAS	N	N	20	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
444	8	5105031807072549	DUSUN MERANGGEN DESA TANGKAS	N	N	24	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
445	9	5105031807072484	DUSUN PEKEN DESA TANGKAS	N	N	30	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
446	368	5105031807072438	DUSUN TUSAN DESA TANGKAS	N	N	15	2017-06-07 15:24:36	2017-06-07 15:54:29	I	0
448	368	5105031807072438	DUSUN AMBENGAN DESA TANGKAS	Y	Y	15	2017-06-07 15:54:29	2017-06-07 16:21:20	U	0
449	368	5105031807072438	DUSUN AMBENGAN DESA TANGKAS	Y	Y	15	2017-06-07 16:21:20	2017-06-07 16:21:20	D	0

Gambar 20. Hasil Proses *Delete Auditing* Untuk Tabel Keluarga

id	id_penduduk	nik	nama	tempat_l...	tanggal_lahir	pendapatan_per_bulan	time_begin	time_end	st_code	flag
1372	1	5105030710700001	I WAYAN NADRASH	TANGKAS	1970-10-07 00:00:00	3000000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1373	5	5105030107600004	I WAYAN TIKA SH	KLUNGKUNG	1960-12-12 00:00:00	1200000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1374	7	5105032004880001	I GEDE JUSTIKA KORI	KLUNGKUNG	1988-04-20 00:00:00	1500000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1375	8	5105030610910001	I MADE DWIJA PARAMARTHA KORI	KLUNGKUNG	1991-10-06 00:00:00	1500000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1376	9	5105030107380060	I KETUT WANDRIS	TANGKAS	1938-12-31 00:00:00	2000000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1377	10	5105034107380087	NI WAYAN ASIH	TANGKAS	1938-12-31 00:00:00	500000	2017-06-07 15:22:54	0000-00-00 00:00:00	I	1
1378	11	5115133112400089	I NENGAH KAYUN	TANGKAS	1940-12-31 00:00:00	2500000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1379	13	5105033112470074	I NENGAH TISTA	TANGKAS	1947-12-31 00:00:00	1500000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1380	14	5105037112510098	NI WAYAN PURNI	GELGEL	1970-01-01 00:00:00	1300000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1381	15	5105033112720022	I KETUT SUCANA	TANGKAS	1972-12-31 00:00:00	1700000	2017-06-07 15:22:55	0000-00-00 00:00:00	I	1
1382	1340	5105033112728742	I MADE BELAYU	TANGKAS	1988-06-14 00:00:00	2000000	2017-06-07 15:27:04	2017-06-07 16:06:33	I	0
1383	1340	5105033112728742	I MADE BELAYU	TANGKAS	1988-06-14 00:00:00	3000000	2017-06-07 16:06:33	2017-06-07 16:22:18	U	0
1384	1340	5105033112728742	I MADE BELAYU	TANGKAS	1988-06-14 00:00:00	3000000	2017-06-07 16:22:18	2017-06-07 16:22:18	D	0

Gambar 21. Hasil Proses *Delete Auditing* Untuk Tabel Penduduk

Gambar 20. dan Gambar 21. menunjukkan data hasil *delete* pada tabel keluarga dengan "id_keluarga" = "368" dan pada tabel penduduk dengan "id_penduduk" = "1340", tersimpan secara historis sebagai *row* baru didalam tabel *auditing*. Data hasil *delete* pada tabel eksisting menunjukan bahwa data tersebut telah non-aktif dari rekaman data hasil *insert* dan *update* pada proses sebelumnya, yang ditunjukkan dengan *field* "flag" bernilai "0".

5. Kesimpulan

Berdasarkan penelitian *database auditing* yang telah di uji coba diatas, dapat disimpulkan bahwa tabel *auditing* seharusnya dipisahkan dari tabel operasional. Hal ini dilakukan dengan tujuan untuk memisahkan beban kerja analisa dari beban kerja transaksi. Mesin DBMS (*Database Management System*) dapat menjalankan *query auditing* lebih cepat ketika tabel *auditing* terpisah dari tabel operasional daripada menjalankan *auditing* terhadap satu tabel besar yang tergabung didalam sistem operasional. Selain itu, DBA (*Database Administrator*) dapat mengelola DBMS menjadi lebih mudah. Memilih desain yang tepat akan mencegah terjadinya penurunan kinerja mesin *database*, mengurangi redundansi data, menghemat penyimpanan, dan menyederhanakan *query auditing*. Hasil dari *auditing* dapat dipelihara untuk keperluan analisis tentang pola akses dan modifikasi terhadap suatu data pada *database* oleh DBA.

Daftar Pustaka

[1] W. Wisswani, "Penerapan Hybrid Slowly Change Dimension Untuk Nearly Realtime Data Warehouse," *Lontar Komput.*, vol. 4, no. 1, 2013.
[2] S. A, M. Sukarsa, and W. B, "Pembentukan Data Mart Menggunakan Metode Generalization," *Lontar Komput.*, vol. 7, no. 3, 2016.
[3] N. Waraporn, "Database Auditing Design on Historical Data," *Proc. Second Int. Symp. Netw. Netw. Secur.*, 2010.
[4] W. Lu and G. Miklau, "Auditing a Database Under Retention Restrictions," *IEEE Int. Conf. Data Eng.*, 2009.
[5] C. Mullins, "Database Auditing Capabilities for Compliance and Security," *The Data Administration Newsletter*, 2008. [Online]. Available: http://tdan.com/database-auditing-capabilities-for-compliance-and-security/8135.
[6] R. Gudakesa, M. Sukarsa, and A. Sasmita, "Two-Ways Database Synchronization In Homogeneous DBMS Using Audit Log Approach," *J. Theor. Appl. Inf. Technol.*, vol. 65, no. 3, 2014.
[7] L. Yang, "Teaching Database Security and Auditing," *Proc. 40th ACM Tech. Sympo-sium Comput. Sci. Educ.*, 2009.
[8] M. C. Muray, "Database Security: What Students Need to Know," *J. Inf. Technol. Educ. Innov. Pract.*, 2010.
[9] H. Surya, "Rancang Bangun Aplikasi Sinkronisasi Database Dua Arah Pada DBMS Homogen Dengan Pendekatan Binary Log," Universitas Udayana, 2014.
[10] M. C. Mazilu, "Database Replication," *Database Sist. J.*, vol. 1, no. 2, 2010.